Mrs. Archana Ajay Nawandhar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.1381-1385

3D Graphics Texture Compression And Its Recent Trends.

***Mrs. Archana Ajay Nawandhar,**
*(Department Of Telecommunication Engineering, CMRIT, Bangalore.)

ABSTRACT

Texture compression is a specialized form of image compression designed for storing texture maps in 3D computer graphics rendering systems. Texture compression is an important technique in graphics processing units (GPUs) for saving memory bandwidth. Unlike conventional image compression algorithms, texture compression algorithms are optimized for random access. Therefore, the variable-rate sequential coding used in many image compression systems cannot be employed, whereas lossy, fixed-rate block-based approaches are mainly used for texture compression algorithms. Secondly it is highly desirable to be able to render directly from the compressed texture data. Therefore in order not to impact rendering performance, decompression must be fast. Along with these requirements high image quality and compression ratio both are needed. This paper gives an overview and comparison between different types of 3-D graphics texture compression standards and algorithms available.

Keywords – **compression, texture, compression ratio, rendering, graphics, texel.**

I. INTRODUCTION

Texturing plays an important role in a graphics pipeline, and the applications of texture data have become much wider in recent graphics systems. The process of applying texture image to the surface of a polygon is texture mapping. An illustration of texture mapping is shown in Fig.(1), where a handmade paper texture is mapped onto a triangle. In the texture mapping of rendering pipelines, two spaces exist: pixel space and texture space. For each fragment (x, y) of the triangle, there is a corresponding texel (s, t) in the texture space. Since s and t are usually not integer numbers, a texture filter–such as point sampling (nearest neighbor) and bilinear filter [1] is usually required to interpolate a fragment's texel value. Texture elements are usually expressed using RGBA tuples, among which RGB stand for color and A means transparency α. Therefore texture compression schemes generally handle both color channels and alpha channel. Texture maps can be viewed as images and they can also be used to store a lot of parameters for rendering, such as transparency, reflectivity, and bumpiness, which greatly increases the reality of the virtual objects and visual effects. To achieve better image quality by considering level of details (LOD), mipmapping textures

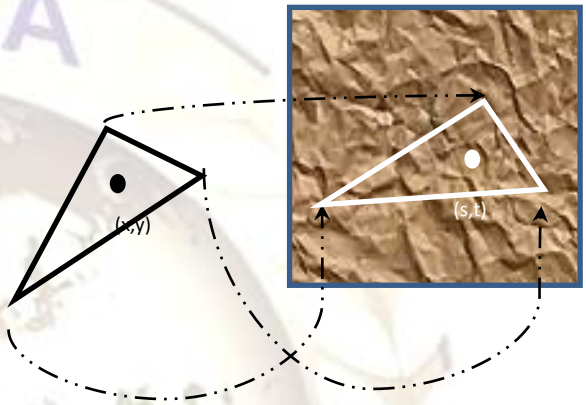

Fig. (1) consisting of multi-scale texture maps are also designed.

When the resolutions of the pixel space and texture space are different, texture mapping with point sampling (nearest neighbour) and bilinear interpolation usually causes aliasing effects. The most popular method for anti-aliasing texturing is trilinear interpolation applied in a mipmapping texture [2], when the mipmapping method is used; the original texture is augmented with a set of smaller versions of the texture. The texture (level zero) is downsampled to a quarter of the original area, with each new texel value often computed as the average four neighbour texels in the level zero texture. The reduction is performed recursively until the last level texture, where only one texel exists. Fig. (2) shows the mipmapping process [5].

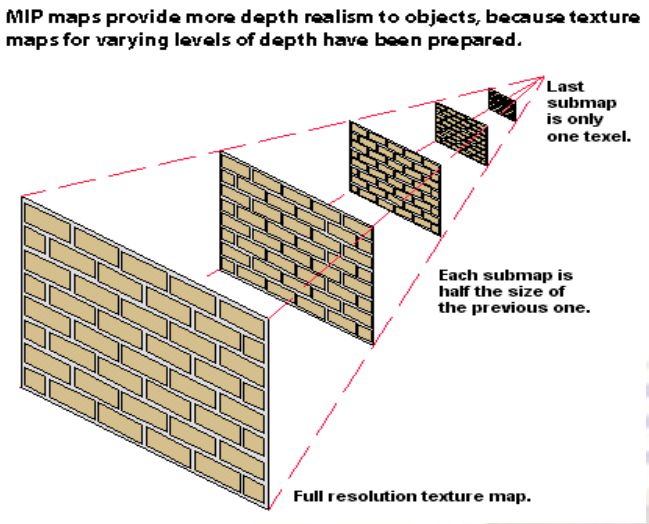

Fig. (2)

However, they also lead to large memory bandwidth requirements. Consequently, in modern graphics processing unit (GPU) hardware architecture the texture unit is a dedicated module: it reads large amount of data from the off-chip texture buffer and sends the required data to the unified shader array after some filtering operations.

In this paper different texture mapping algorithms like S3TC's DXT1-7, ASTC, PVRTC, ATI's 3DC are discussed.

II. S3 TEXTURE COMPRESSION (S3TC)

Sometimes also called DXTn or DXTC; is a group of related lossy texture compression algorithms originally developed by Iourcha et al. of S3 Graphics, Ltd.[4] for use in their Savage 3D computer graphics accelerator. The method of compression is similar to the Color Cell Compression. Color Cell Compression is an early lossy image compression algorithm first described by Campbell et al. in 1986.[1] It is a variant of Block Truncation Coding. The encoding process works on small blocks of pixels. For each block, it first partitions the pixels in that block into two sets based on their luminance values, then generates representative colour values for each of these sets, and a bitmap that specifies which pixels belong to which set. The two colour values and the bitmap for each block are then output directly without any further quantization or entropy coding.

The decoding process is simple; each pixel of an output block is generated by choosing one of the two representative colours for that block, based on that block's bitmap. It had the advantage of very simple decompression and fast random access into the compressed image, and it can be regarded as a forerunner of modern texture compression algorithms.

Table (1) . S3TC Format Comparison

FOUR CC	DX 10 Name	Description	Alpha premultiplied?	Compression ratio	Texture Type
DXT1	BC1	1-bit Alpha / Opaque	N/A	6:1(for 24 bit source image)	Simple non-alpha
DXT2	(none)	Explicit alpha	Yes	4:1	Sharp alpha
DXT3	BC2	Explicit alpha	No	4:1	Sharp alpha
DXT4	(none)	Interpolated alpha	Yes	4:1	Gradient alpha
DXT5	BC3	Interpolated alpha	No	4:1	Gradient alpha

S3TC supports fixed-rate data compression coupled with the single memory access. There are five variations of the S3TC algorithm named DXT1 through DXT5, each designed for specific types of image data. All convert a 4×4 block of pixels to a 64-bit or 128-bit quantity, resulting in compression ratios of 6:1 with 24-bit RGB input data or 4:1 with 32-bit RGBA input data. Since its a lossy compression algorithm, it results in an image quality degradation, an effect which is minimized by the ability to increase texture resolutions while maintaining the same memory requirements. Hand-drawn cartoon-like images do not compress well, nor do normal map data, both of which usually generate artifacts.

The only really important ones are DXT1, DXT3, and DXT5. They all handle colour compression in the same way, but the key difference is how they handle the alpha channel. DXT1 has a binary alpha, and is basically unsuitable for anything that will be rendered with alpha blending (it might be ok for simple alpha testing). DXT3 stores an uncompressed 4-bit alpha for each pixel. This means that the colours are all in the right place, but there are only 16 levels to choose from, causing severe banding artifacts on long gradients. DXT5 stores an interpolated 4-bit alpha. This means that there are the same kind of block compression artifacts as in the colour channel, but smooth gradients look pretty good. It's safe to say that DXT5 alpha channels look the best for most images. If there's no alpha channel, we should use DXT1, because it's half as big as DXT5. If there are only sharp edges, DXT3 is actually slightly more accurate than DXT5. Fig 3 shows alpha channel of a smoke sprite compressed with each type:

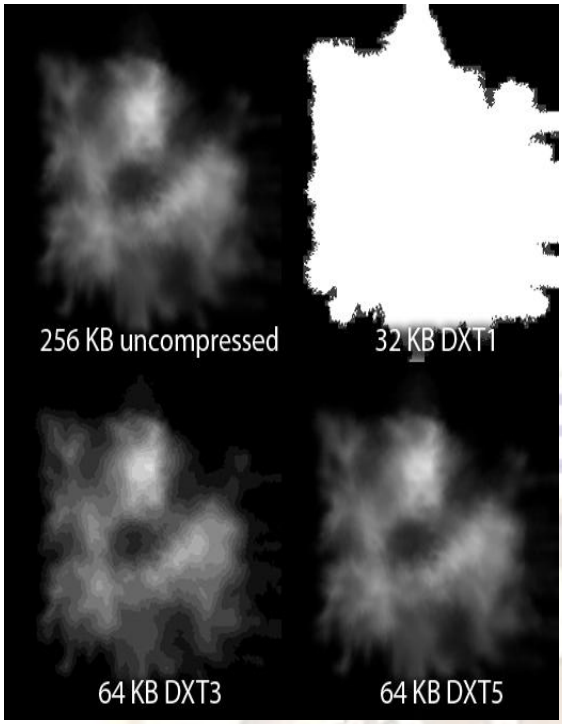

Fig 3

III. ATI'S 3DC COMPRESSION ALGORITHM

ATI's 3Dc compression algorithm is a modification of DXT5 designed to overcome S3TC's shortcomings with regard to normal maps. Normal maps are special textures that are used to add detail to 3D surfaces. They are an extension of earlier "bump map" textures, which contained per-pixel height values and were used to create the appearance of bumpiness on otherwise smooth surfaces. Normal maps contain more detailed surface information, allowing them to represent much more complex shapes. ATI's 3Dc technology is designed to allow game developers to pack more detail into real time 3D images than ever before, making 3Dc a key enabler of the HD Gaming vision. DXTC & S3TC are ineffective at compressing normal maps. They tend to have trouble capturing the small edges and subtle curvature that normal maps are designed to capture, and they also introduce unsightly block artifacts fig. 4. Because normal maps are used to capture light reflections and realistic surface highlights, these problems are amplified relative to their impact on color textures. The results are sufficiently poor that game artists and developers would rather not use normal maps at all on most surfaces, and instead limit themselves to lower resolution maps on selected parts of the rendered scene.

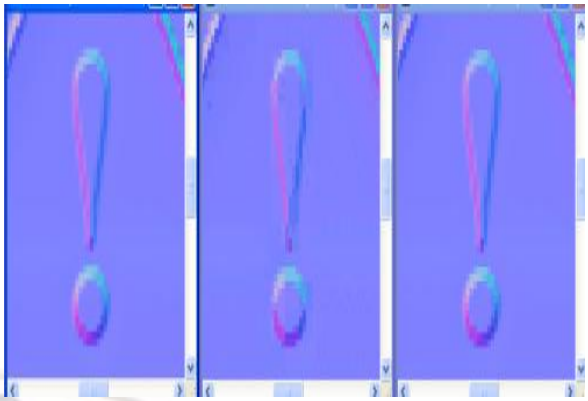

Fig 4: uncompressed image (leftmost); DXT5 compressed image (middle); 3Dc compressed image (rightmost).

3Dc is a block-based compression technique. It breaks a texture map up into 4x4 blocks containing 16 values each. These values must consist of two components. Each component is compressed separately. A maximum and minimum value is determined for each block, and these are stored as 8-bit values. A set of six intermediate values are then calculated, spaced equally between the minimum and the maximum. This gives a total of eight values that each component can take within a block. Each component is assigned a 3-bit index corresponding to whichever of these values is closest to its original value. The resulting compressed blocks consist of four 8-bit values and thirty-two 3-bit values, for a total of 128 bits. Since the original blocks consisted of sixteen 32-bit values, for a total of 512 bits, this represents a compression ratio of 4:1. If the original values were 16-bit rather than 32-bit, then a compression ratio of 2:1 can still be achieved. Using 3Dc to compress normal maps requires an additional step. This is because each value in a normal map is actually a 3D vector, consisting of 3 components (x, y & z). These values must be reduced to 2-component values in order to work with 3Dc. Fortunately, this can be handled in a simple way by assuming that all of the normal vectors have a length of 1. Given the values of two components of a vector, the value of the third component can be found using the following mathematical relationship: $z = \sqrt{(1 - (x2+y2))}$. This formula can be implemented using just a couple of pixel shader instructions. Thus 3Dc is an exciting new compression technology designed to bring out fine details in games while minimizing memory usage. It is the first compression technique optimized to work with normal maps, which allow fine per-pixel control over how light reflects from a textured surface. With up to 4:1 compression possible, this means game designers can now include up to 4x the detail without changing the amount of graphics memory required and without impacting performance.

III. VARIABLE BIT RATE TEXTURE COMPRESSION.

All the DXT1-5 are fixed bit rate compression techniques and they inevitably has some blocks that are compressed too much, leading to artifacts, and some blocks that could be compressed more, leading to larger files.to overcome this variable bit rate compression techniques are being developed. Some of them are DXT6, DXT7, and ASTC. The VBR texture compression algorithm [6] are designed for fast GPU decompression. A high compression rate algorithm that can be decompressed on the GPU allows a vast increase in on-GPU texture storage. Even as future GPUs move to unified memory, memory will still limit texture capacity, and fast GPU decompression still increases total texture capacity. Unlike most image compression algorithms, which only reconstruct an image at a single resolution, VBR algorithm reconstructs the entire MIP chain, avoiding the significant overhead of compressing each MIP level independently, while allowing independent selection of MIP filter or artistic tailoring of the MIP levels.

Adaptive Scalable Texture Compression (ASTC) is a lossy block-based texture compression algorithm [3] developed by Jørn Nystad et al. of ARM Ltd. The stated primary design goal for ASTC is to enable content developers to have better control over the space/quality tradeoff inherent in any lossy compression scheme. With ASTC, the ratio between adjacent bit rates is of the order of 25%, making it less expensive to increase quality for a given texture.

Encoding different assets often requires different color formats. ASTC allows a wide choice of input formats, including luminance-only, luminance-alpha, RGB, RGBA, and modes optimized for surface normals. The designer can thus choose the optimal format without having to support multiple different compression schemes. The choices of bit rate and color format do not constrain each other, so that it possible to choose from a large number of combinations. Despite this flexibility, ASTC achieves better peak signal-to-noise ratios than PVRTC, S3TC, and ETC2 when measured at 2 and 3.56 bits per texel. For HDR textures, it produces results comparable to BC6H at 8 bits per texel.

ASTC textures are compressed using a fixed block size of 128 bits, but with a variable block footprint ranging from 4x4 texels up to 12x12 texels. The available bit rates thus range from 8 bits per texel down to 0.89 bits per texel, with fine steps in between.

IV. PVRTC AND PVRTC2

These are from the family of lossy, fixed-rate texture compression formats used in PowerVR's MBX (PVRTC only), SGX and Rogue technologies. These differ from block-based texture formats such as S3TC and Ericsson Texture Compression (ETC) in that the compressed image is represented by two lower resolution images which are bilinearly upscaled and then blended according to low precision, per-pixel weights. They also differ in that they support ARGB data in both 4-bpp and 2-bpp modes. PVRTC is the compressed texture format used in all generations of the iPhone, iPod Touch, and iPad.

IV. CONCLUSION

In computer graphics achieving high visual quality typically requires high-resolution textures. However the desire for increasing texture resolution conflicts with the limited amount of graphics memory and available. Memory bandwidth is the most important aspect of graphics system performance today. Especially for embedded systems increasing the memory bandwidth may not be an option. Texture compression can help to achieve higher graphics quality with given memory and bandwidth without degrading quality too much. DXT compression is a lossy texture compression algorithm that can reduce texture storage requirements and decrease texture bandwidth. The DXT compressions are good to use on decal texture images, especially those images that are high in resolution BUT does not give good results with normal maps. Whereas ATI's 3Dc compression technique can be used effectively to compress normal map data or to compress multiple pieces of data into a single texture. This format compresses the images while retaining the highest level of detail in it. Which texture format and compression technique needs to be used will depend on the type of images and the target application requirements. ASTC provides better peak signal-to-noise ratios as compared to other compression formats. PVRTC compression algorithm is used for embedded systems. It tries to take advantage of the correlation of texel position and color, used in PDAs and smartphones..

REFERENCES

Journal Papers:

[1] Chih-Hao Sun, You-Ming Tsao, and Shao-Yi Chien, Member Graphics Processing Units;, IEEE;"High-Quality Mipmapping Texture compression With Alpha Maps for";IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 11, NO. 4, JUNE 2009

[2] Campbell, G.; Defanti, T. A.; Frederiksen, J.; Joyce, S. A.; Leske, L. A. (1986). "Two bit/pixel full color encoding". Proceedings of the 13th annual conference on Computer graphics and interactive techniques -

SIGGRAPH '86. pp. 215. doi:10.1145/ 15922.15910. ISBN 0897911962.

[3] US 5956431 "Fixed-rate block-based image compression with inferred pixel values"

[4] Yifei Jiang∗, Mindan Gui†, Dongdong Lu∗, Yuanchao Xu‡§ ∗Shanghai High Performance IC Design Center, Shanghai, China †Department of Electronic & Information Engineering, Wuxi South Ocean College, Wuxi, China ‡Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China §College of Information S3 Texture Compression; en.wikipedia.org/wiki/ S3_Texture_Compression Engineering, Capital Normal University, Beijing, China jiangyifei01@gmail.com

[5] Adaptive Scalable Texture compression (ASTC), en.wikipedia.org/wiki/ Adaptive_ Scalable_ Texture_compression

[6] Texture Compression with Variable Data Formats 2012 IEEE 12th International Conference on Computer and Information TechnologyM Ozaki, Y. Adachi, Y. Iwahori, and N. Ishii, Application of fuzzy theory to writer recognition of Chinese characters, *International Journal of Modelling and Simulation, 18(2),* 1998, 112-116.

Books:

[7] Tomas Akenine-Moller, Eric Haines, Naty Hoffman, *Real time graphics rendering* [3e A K Peters LTD: Natick, MA] .